High Performance Enterprise Computing with ACML 2.0

Tim Wilkens Ph.D.
Member of Technical Staff
tim.wilkens@amd.com

Acknowledgments

Work carried out in collaboration with the **N**umerical **A**lgorithms **G**roup (**NAG**):

Lawrence Mulholland
Mick Pont
Stef Salvini
Ed Smyth
Themos Tsikas

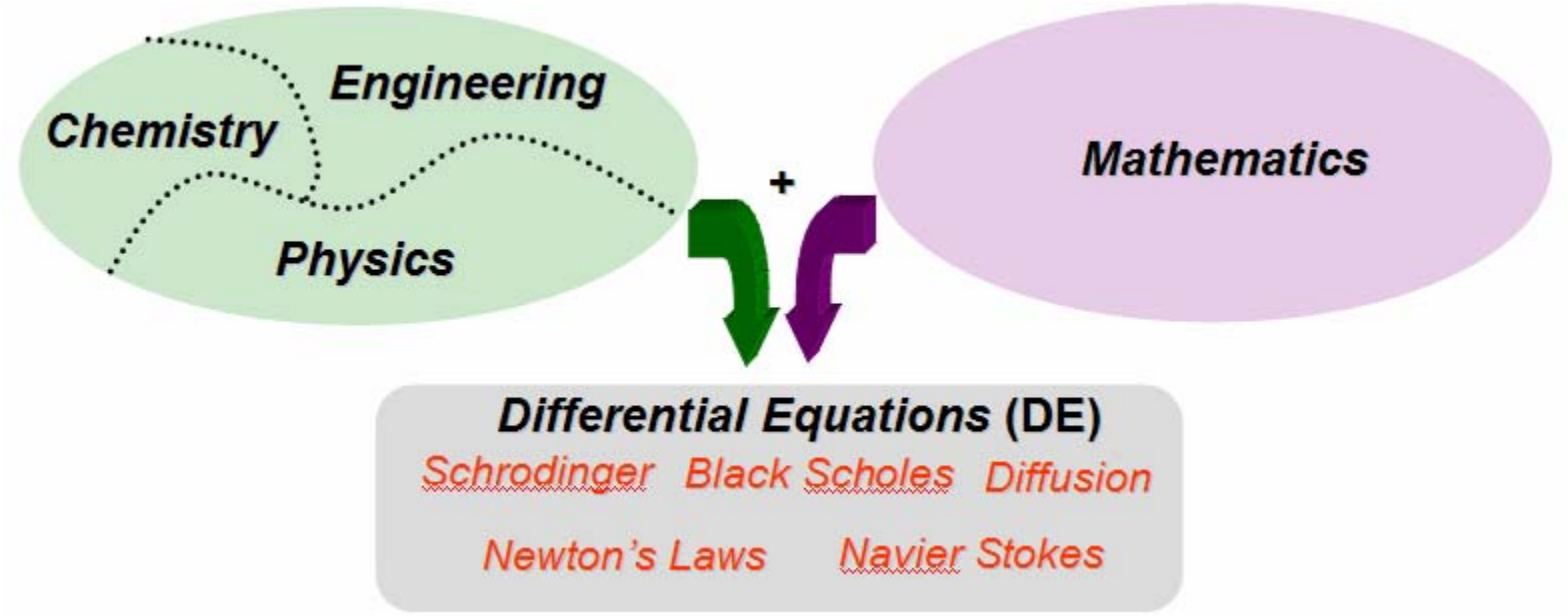
NAG also provides its numerical libraries for AMD64 technology which leverage ACML.

ACML Project Manager ***Chip Freitag*** for his tools and OS support

Agenda

- Why Build a High Performance Math Library
- Overview of AMD Opteron™ Processor
- BLAS, LAPACK and FFT Performance
- Summary and Closing Points

What are Math Libraries Good For?
Continuous vs Discrete



- DEs exist only in continuous space
- Few analytic DE solutions exist

BUT, if spatial and time dimensions are made discrete…

…DE become Matrix Equations which can be solved using Linear Algebra

Components of ACML
BLAS, LAPACK, FFTs

- **Linear Algebra (LA)**
  - **B**asic **L**inear **A**lgebra **S**ubroutines (**BLAS**)
    - Level 1 (vector-vector operations)
    - Level 2 (matrix-vector operations)
    - Level 3 (matrix-matrix operations)
    - Routines involving sparse vectors
  - **L**inear **A**lgebra **PACK**age **(LAPACK)**
    - leverage BLAS to perform complex operations
    - 28 Threaded LAPACK routines
- **Fast Fourier Transforms** (**FFTs**)
  - 1D, 2D, single, double, r-r, r-c, c-r, c-c support
- **C and Fortran interfaces**

Linear Algebra PACKage (LAPACK)
Solving Matrix Equations

- **Matrix Equation Solvers (2 step process)**
  - Matrix Decomposition (LU, QR, Cholesky, SVD)
  - Backward Substitution

- **Uses:**
  - Least Squares Minimization
  - Eigenvalue / Eigenvector Solutions
  - Determining Condition numbers of problems

ACML LAPACK leverages **Level 3 BLAS** routines,
performance gated by **FPU throughput *not* memory bandwidth**

Driving Enterprise Business
Market Segments Using BLAS/LAPACK

Structural Analysis

GE, Raytheon, Boeing, Lockheed, Motorola, Fiat, Toyota, Pratt & Whitney, Dupont, Rolls Royce, Corning, General Dynamics, GTE

Computational Fluid Dynamics

Boeing, Ferrari, Raytheon, Lockheed, Daimler, Morton Thiokol, AMD, Martin Marietta, Ducati, Renault

Crash Analysis

NASA, Boeing, Volvo, Mitsubishi, Ferrari, Volkswagen, Airbus, GM, Ford, Daimler, Honda

Digital Signal Processing

NSA, DEA, CIA, Texas Instruments, AT&T, Sprint, MCI

Oil and Gas

Shell, BP, Total, Petrobras, Halliburton, ChevronTexaco, ExxonMobil, Aramco

Education Defense

PNNL, LANL, LLNL, ORNL, NCSA, ANL, SNL, BNL, FNAL, NERSC

Materials Science Biology

Eli Lilly, Bristol Meyers, Dow Chemical, DuPont, Union Carbide, Pfizer, Genentech, Genencor, Accelrys, Incyte Genomics

Financial Analysis

NumeriX, Palisade, MathWorks, Wolfram Research, Goldman Sachs, Morgan Stanley, JP Morgan, Salomon Brothers

Driving Enterprise Business

Market Segments Using FFTs

Structural Analysis

GE, Raytheon, Boeing, Lockheed, Motorola, Fiat, Toyota, Pratt & Whitney, Dupont, Rolls Royce, Corning, General Dynamics, GTE

Computational Fluid Dynamics

Boeing, Ferrari, Raytheon, Lockheed, Daimler, Morton Thiokol, AMD, Martin Marietta, Ducati, Renault

Crash Analysis

NASA, Boeing, Volvo, Mitsubishi, Ferrari, Volkswagen, Airbus, GM, Ford, Daimler, Honda

Digital Signal Processing

NSA, DEA, CIA, Texas Instruments, AT&T, Sprint, MCI

Oil and Gas

Shell, BP, Total, Petrobas, Halliburton, ChevronTexaco, ExxonMobil, Aramco

Education Defense

PNNL, LANL, LLNL, ORNL, NCSA, ANL, SNL, BNL, FNAL, NERSC

Materials Science Biology

Eli Lilly, Bristol Meyers, Dow Chemical, DuPont, Union Carbide, Pfizer, Genentech, Genencor, Accelrys, Incyte Genomics

Financial Analysis

NumeriX, Palisade, MathWorks, Wolfram Research, Goldman Sachs, Morgan Stanley, JP Morgan, Salomon Brothers

Connecting HPC and you

How HPC impacts our daily lives

- **Gaming – Real World Realism**
  - water surfaces, physics gaming engines
- **Rendered Movies**
  - modeling real clothing surfaces (PDEs)
- **Medical Procedures**
  - **CAT** scan imaging, Cancer Radiation Therapy
- **Airline Flight Schedules**
  - minimizing equations of constraint (fuel, food, time, etc)
- **National Security**
  - voice analysis and authentication, weapons simulation

AMD Core Math Library (ACML)

Assembly and Accuracy

Floating-point Numbers are represented in many formats specified by a set of parameters:

$$x = \pm \underbrace{\left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_{t-1}}{\beta^{t-1}} \right)}_{\text{mantissa}} \underbrace{\beta^e}_{\text{exponent}}$$

$$0 \le d_i \le \beta - 1 \qquad -N + 1 \le e \le N$$

$$i = 0, \ldots, t-1$$

The IEEE standard sets β to 2 to for beneficial reasons

t and ***N*** are dictated by the precision of the number being represented

FP Operation	*t*	*N*
SSE	24	127
SSE2	53	1,023
X87	64	32,767

AMD Core Math Library (ACML)

Assembly usage in ACML

- 3 **ACML** 32-bit binaries
  - supports present and past AMD processors with/without SSE & SSE2
- 1 **ACML** 64-bit binary
  - supports *AMD Athlon™ 64* and *AMD Opteron™* processors

Address Space Supported	Processors Supported	BLAS Assembly Type	FFT Assembly Type
32-bit	AMD Athlon™ Processor	*X87* *X87*	*X87* *X87*
32-bit	AMD Athlon MP AMD Athlon XP Processors	*SSE* *X87*	*SSE* *X87*
32-bit	AMD Opteron™ Processor	*SSE* *X87*	*SSE* *SSE2*
64-bit	AMD Opteron Processor	*SSE* *SSE2*	*SSE* *SSE2*

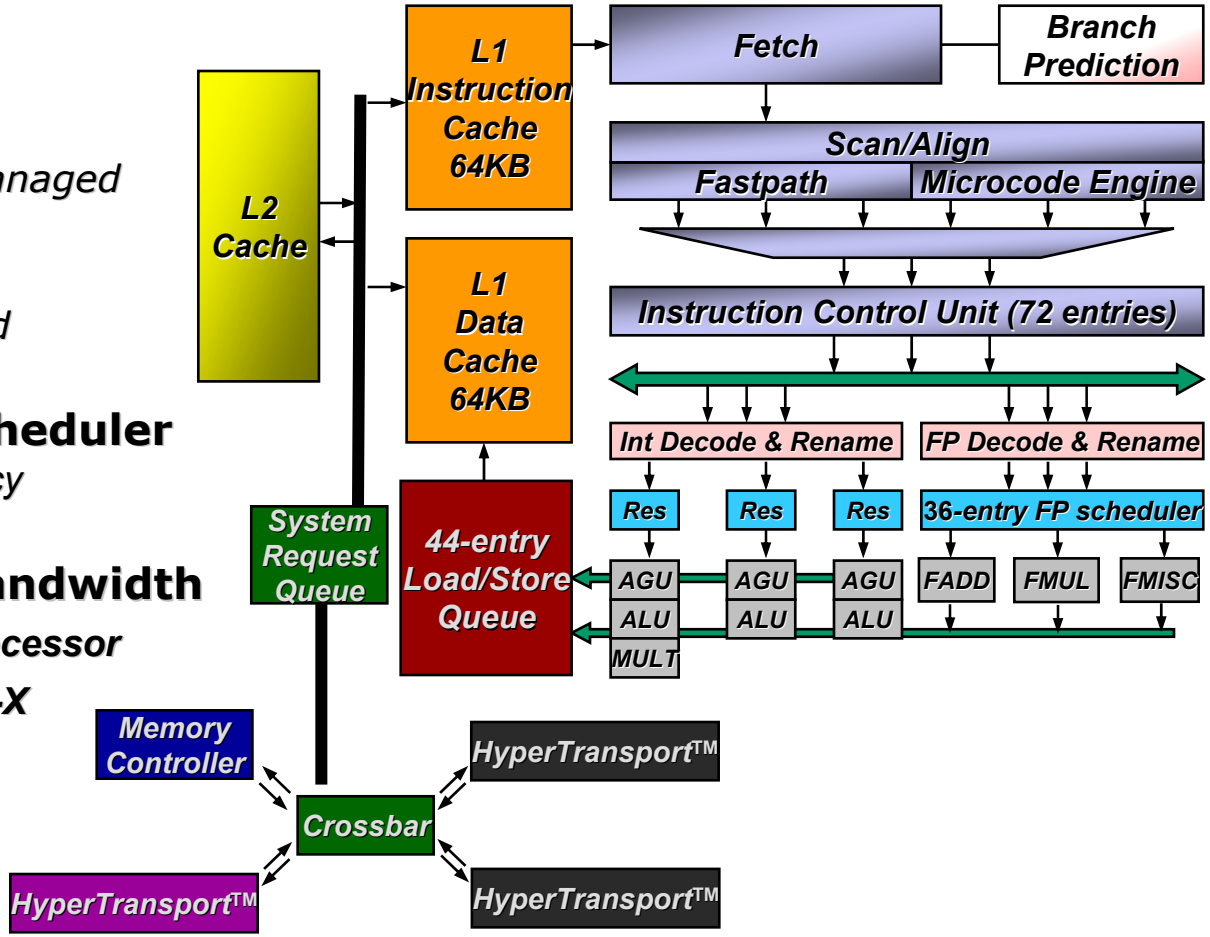
single precision

double precision

The AMD Opteron™ Processor core

The Engine beneath the hood

- **RISC (not CISC) core**
  High Clock Frequency
- **Register Renaming**
  processor vs compiler managed
- **Out of Order Execution**
  Less code tuning required
- **3 FPU Pipelines to Scheduler**
  Greater Scalar FPU Efficiency
- ***up to* 25.6 GB/s of Bandwidth**
  - *up to* ***12.8 GB/s Inter-processor***
  - *up to* ***6.4 GB/s AGP, PCI-X***
  - *up to* ***6.4 GB/s Memory***



AMD Opteron™, Pentium®4 (FPU analysis)

Throughput of SSE, SSE2, x87 Operations

Operation	SSE Scalar	SSE vector	SSE2 scalar	SSE2 vector	X87
Add	1 / cycle	2 / cycle	1 / cycle	1 / cycle	1 / cycle
Multiply	1 / cycle	2 / cycle	1 / cycle	1 / cycle	1 / cycle
Add & Multiply	2 / cycle	4 / cycle	2 / cycle	2 / cycle	2 / cycle

Operation	SSE Scalar	SSE vector	SSE2 scalar	SSE2 vector	X87
Add	1 / 2 cycles	2 / cycle	1 / 2 cycles	1 / cycle	1 / cycle
Multiply	1 / 2 cycles	2 / cycle	1 / 2 cycles	1 / cycle	1 / 2 cycles
Add & Multiply	1 / cycle	4 / cycle	1 / cycle	2 / cycle	1 / cycle

AMD Opteron™, Pentium®4 (ALU Analysis)

Throughput and Latency Comparison

Operation	32-bit	64-bit
ADD/SUB	3 / cycle	3 / cycle
MUL_{signed}	1 / cycle 4 cycle latency	1 / 2 cycles
$MUL_{unsigned}$	1 / cycle 4 cycle latency	1 / 2 cycles
$MOV_{mem,reg}$	2 / cycle	2 / cycle
$MOV_{reg,reg}$	3 / cycle	3 / cycle
XOR/AND/OR	3 / cycle	3 / cycle
Shift/Rotate	3 / cycle	3 / cycle
DIV signed	42 cycle latency	
DIV unsigned	39 cycle latency	
LEA	3 / cycle	3 / cycle

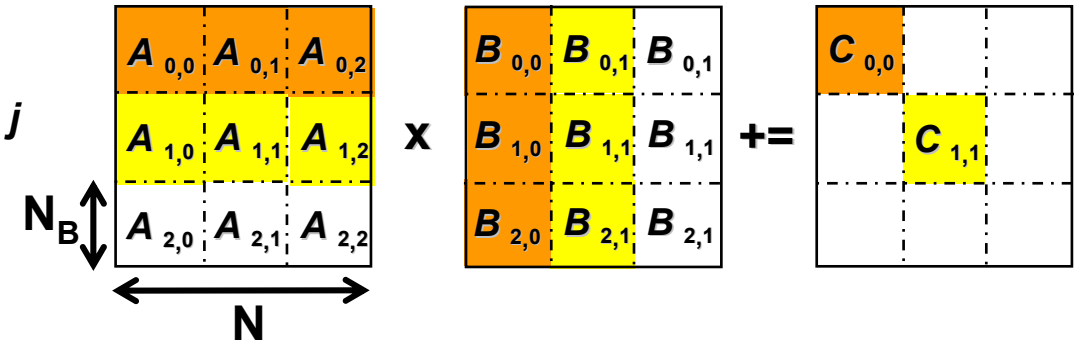
Operation	32-bit	64-bit
ADD/SUB	2 / cycle	NA
MUL_{signed}	1 / cycle 18 cycle latency	NA
$MUL_{unsigned}$	1 / cycle 10 cycle latency	NA
$MOV_{mem,reg}$	2 / cycle	NA
$MOV_{reg,reg}$	2 / cycle	NA
XOR/AND/OR	2 / cycle	NA
Shift/Rotate	2 / cycle	NA
DIV signed	80 cycle latency	NA
DIV unsigned	80 cycle latency	NA
LEA	(2–0.5) / cycle	NA

BLAS Optimization Tactics
Cache Blocking

- Generic **D**ouble precision **GE**neral **M**atrix-**M**ultiply (**DGEMM**) is performance limited by memory bandwidth for large problems

- Alternatively Block A and B into portions that fit into L1 cache and calculate blocks of C in the manner below:

$$C_{i,j} = \sum_{k=0}^{N/N_B} A_{i,k} \times B_{k,j}$$



- ***Advantages***:
  - blocked data ***16-byte aligned*** and ***local to CPU via NUMA***
  - lots of L1 and L2 cache reusage

Provides approximately linear performance scaling with CPU frequency and FPU throughput

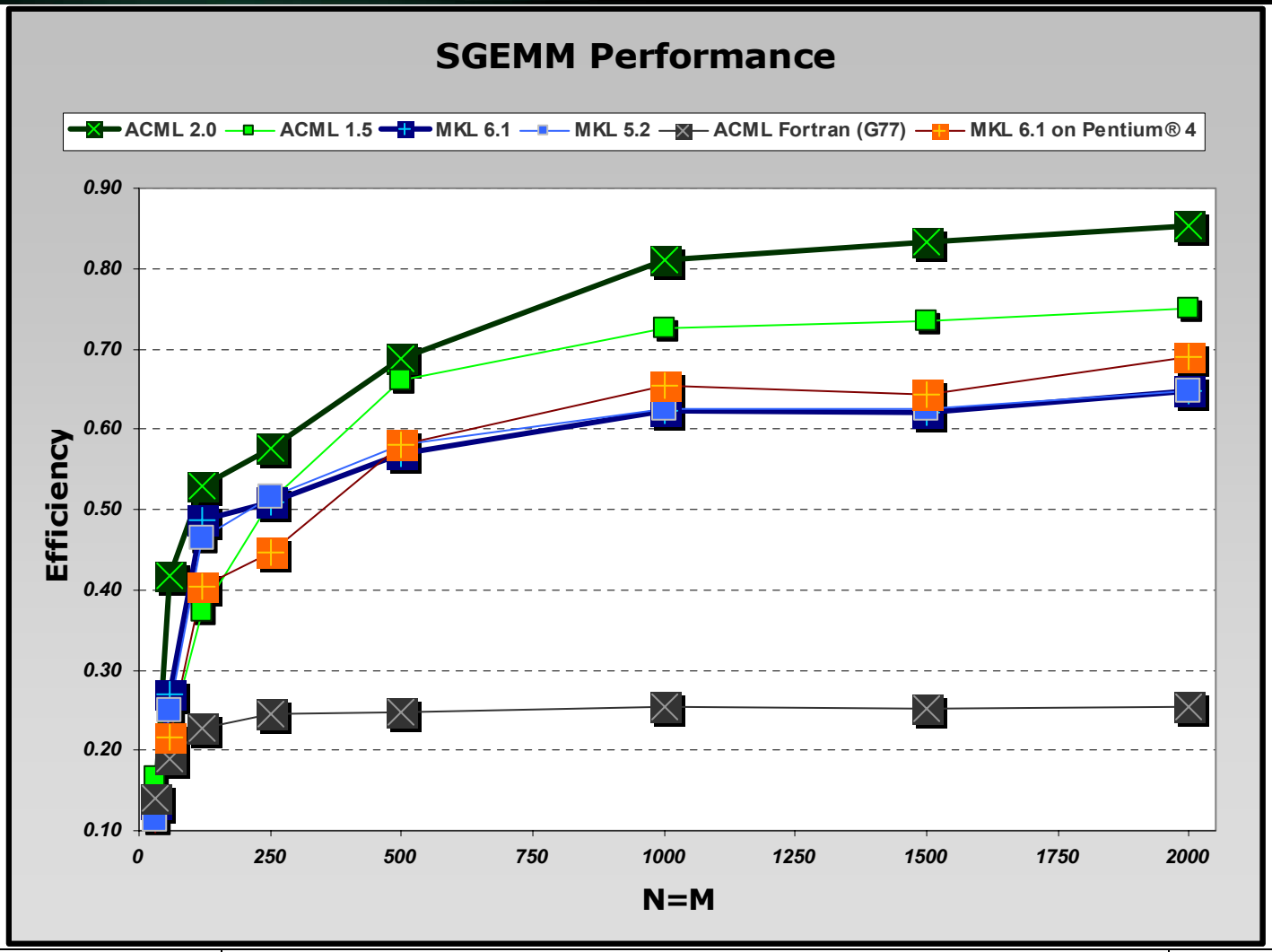
FFT Optimization Tactics

The most difficult task in optimizing ACML

- **FFT**s suffer from **intense register pressure**, **dependency chains**, **instruction set limitations**, **cache associativity**, etc.
- Choose an FFT algorithm that minimizes data shuffling
- Address memory and handle register pressure with the least number of register spills
- Arrange User and Library data in a manner that vectorizes efficiently
- Cache and memory bound problems require different algorithms
- Interoperability of even and odd radices with minimal overhead

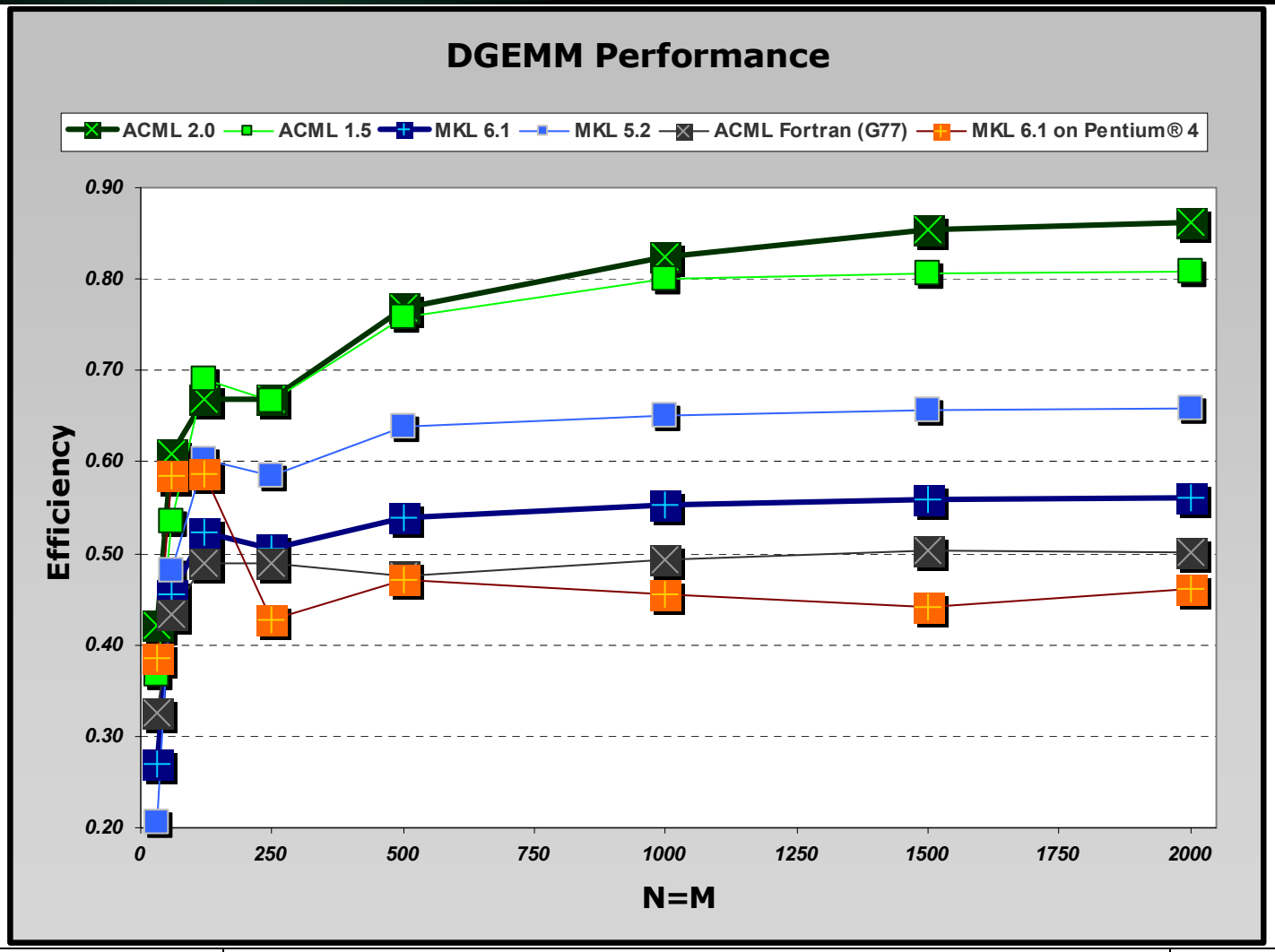
64-bit BLAS Performance

SGEMM (Single Precision General Matrix Multiply)



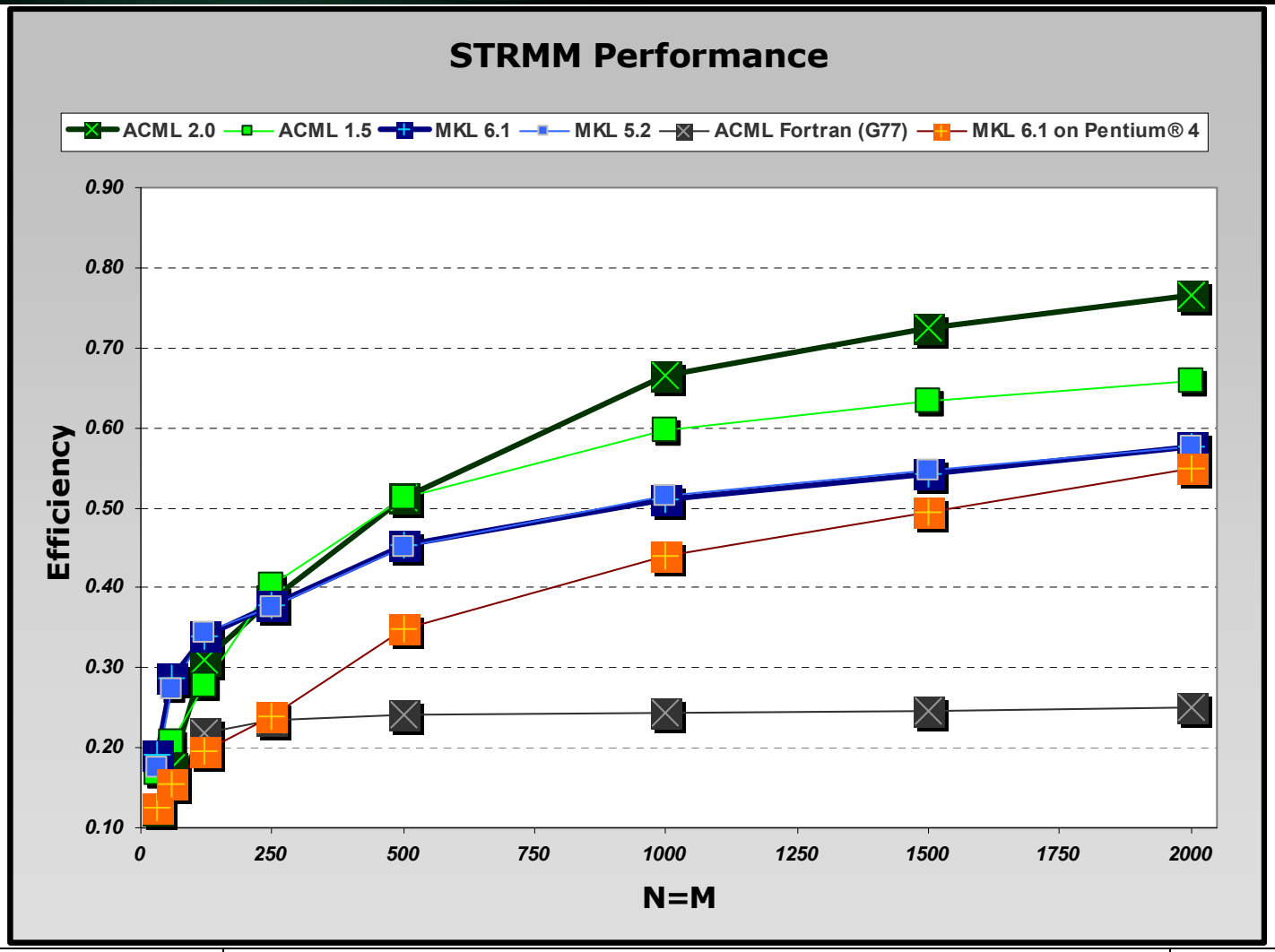
64-bit BLAS Performance

DGEMM (Double Precision General Matrix Multiply)



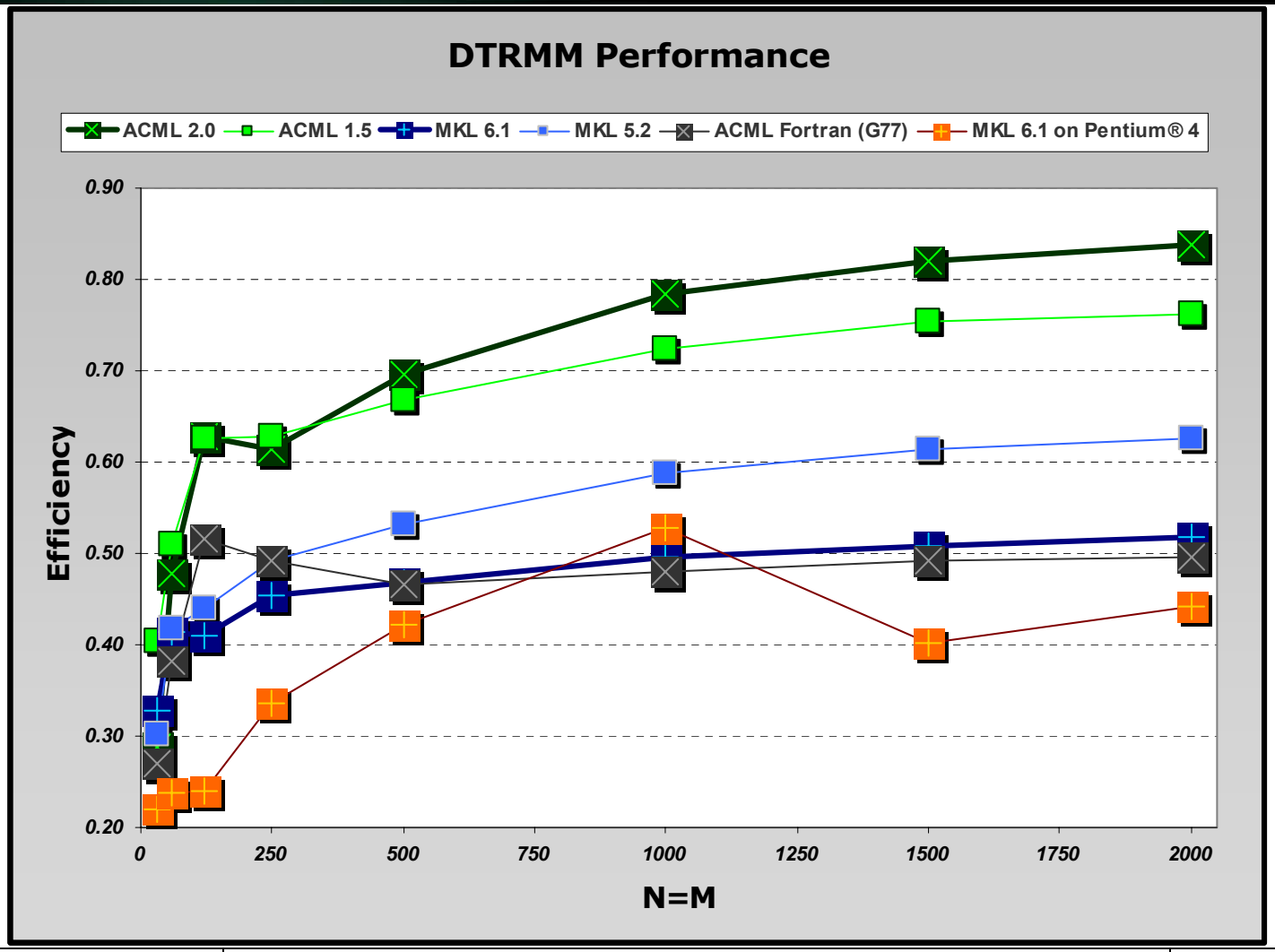
64-bit BLAS Performance

STRMM (Single Precision Triangular Matrix Multiply)



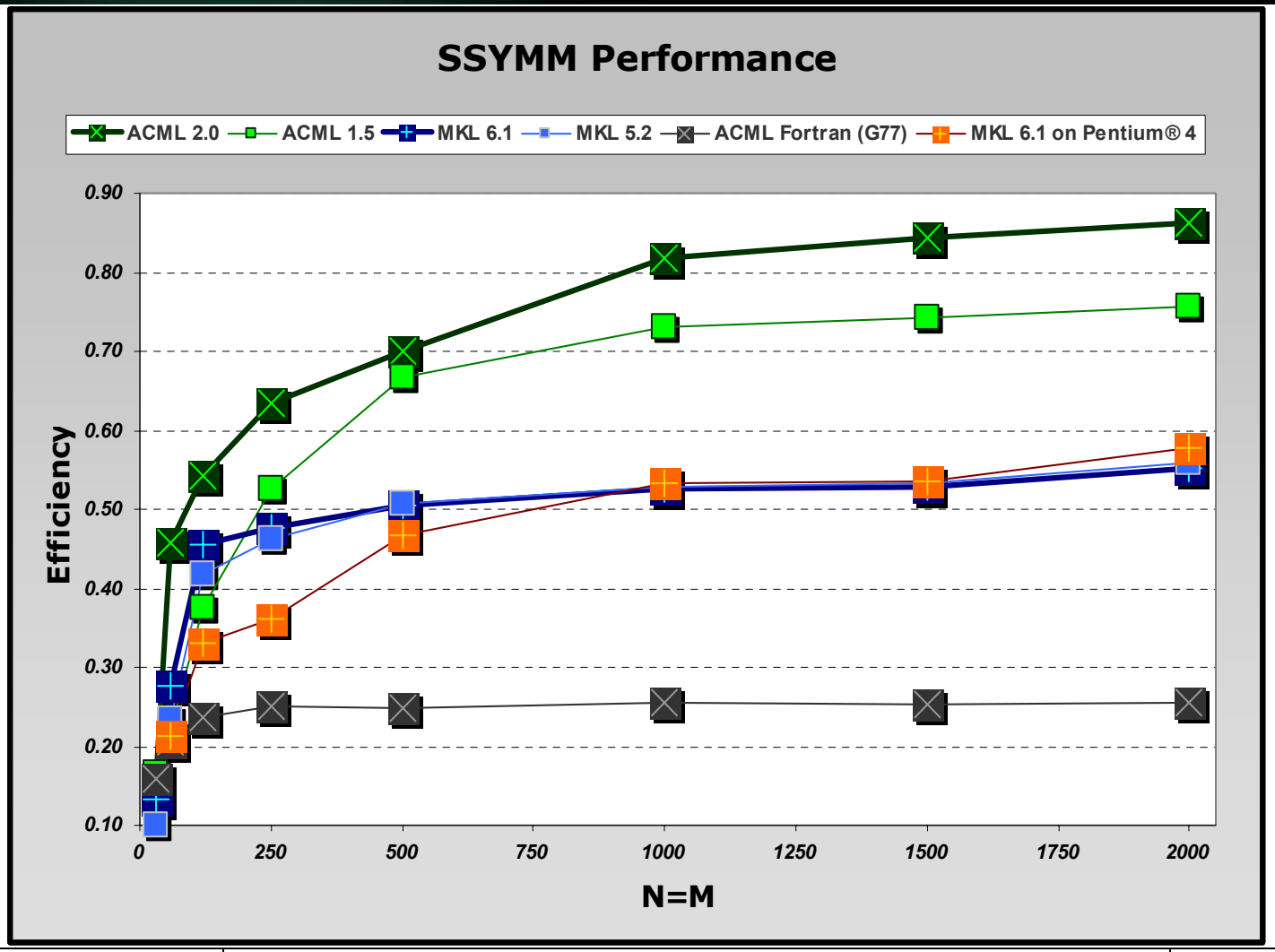
64-bit BLAS Performance

DTRMM (Double Precision Triangular Matrix Multiply)



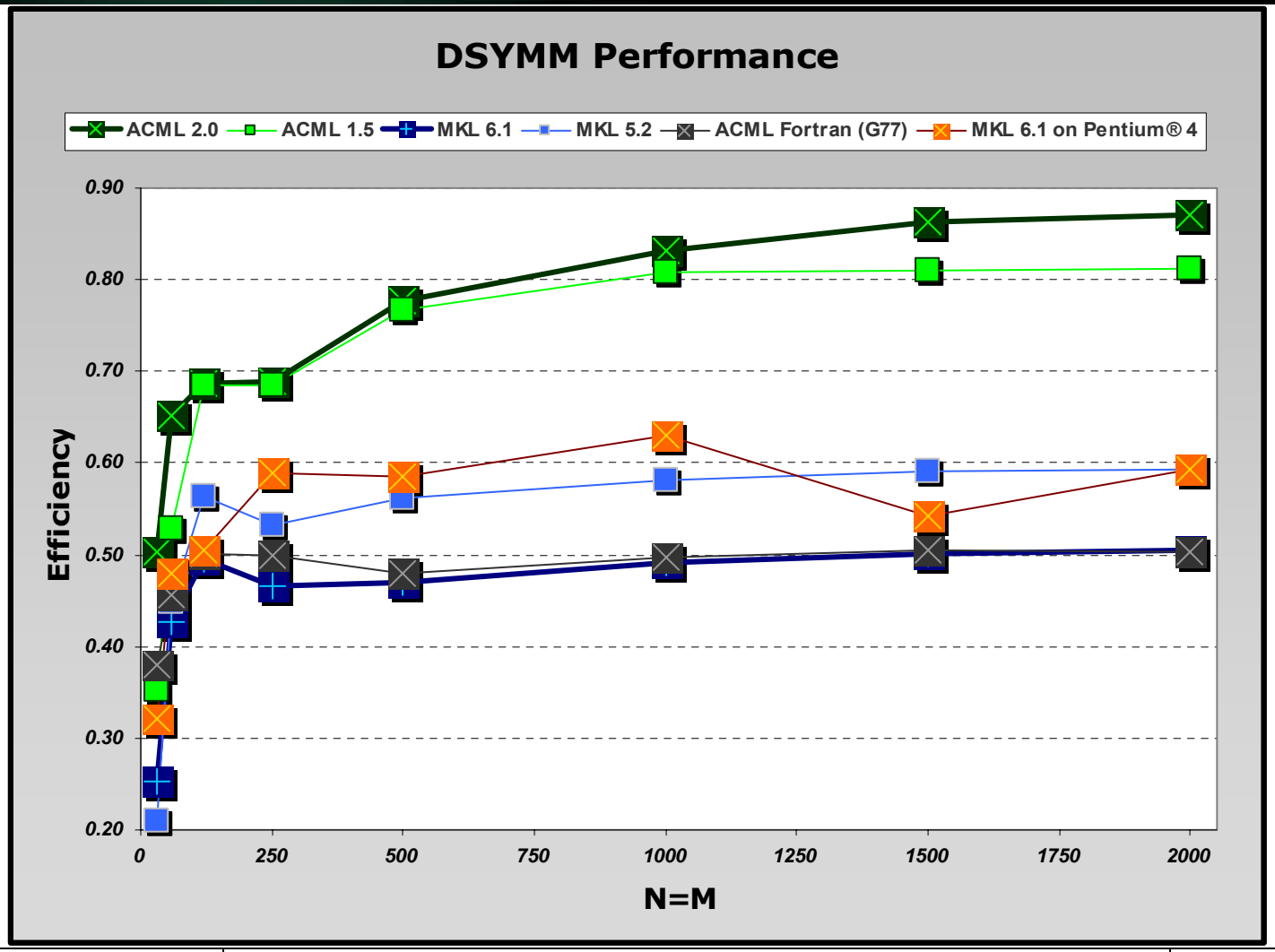
64-bit BLAS Performance

SSYMM (Single Precision Symmetric Matrix Multiply)



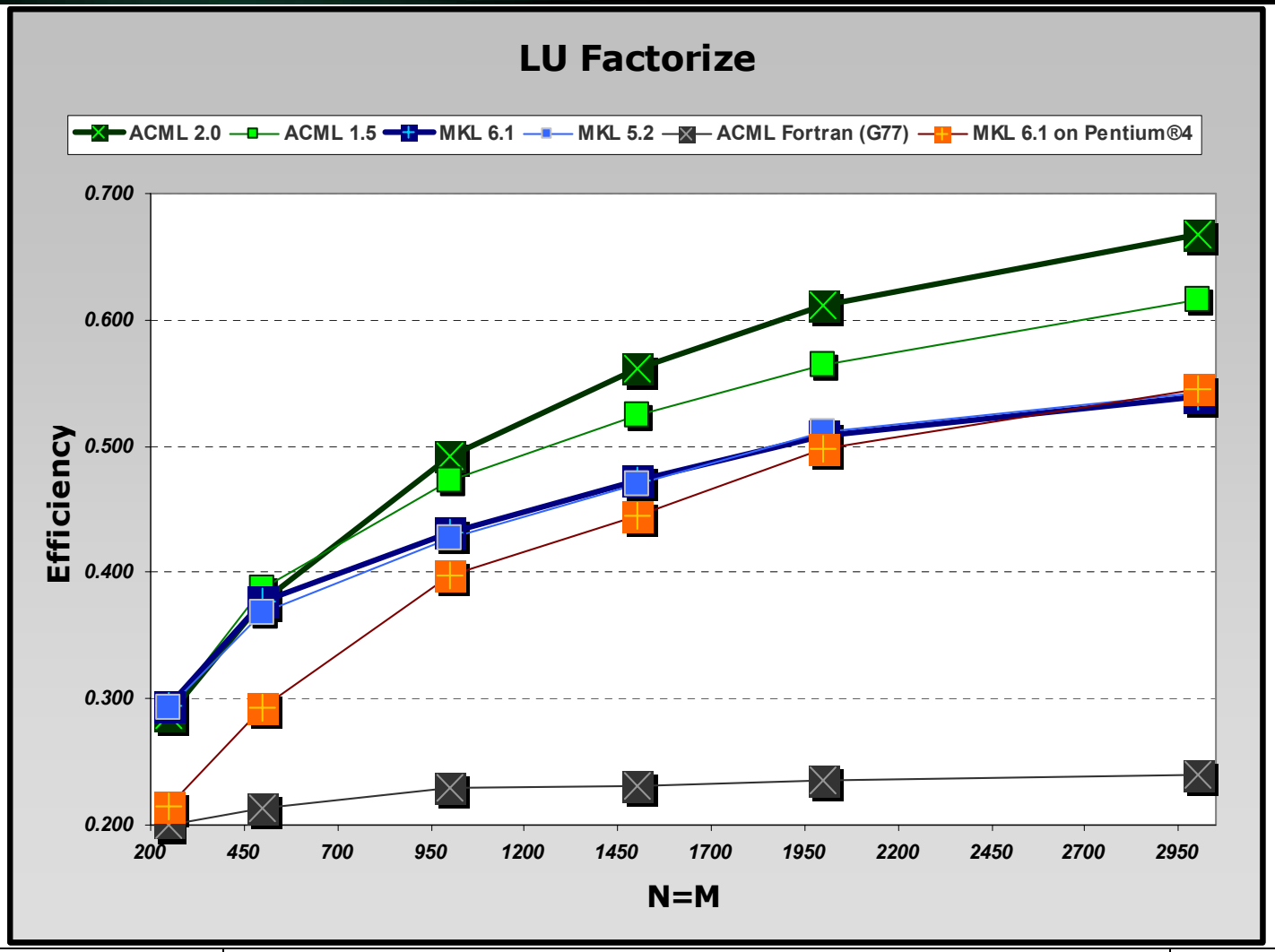
64-bit BLAS Performance

DSYMM (Double Precision Symmetric Matrix Multiply)



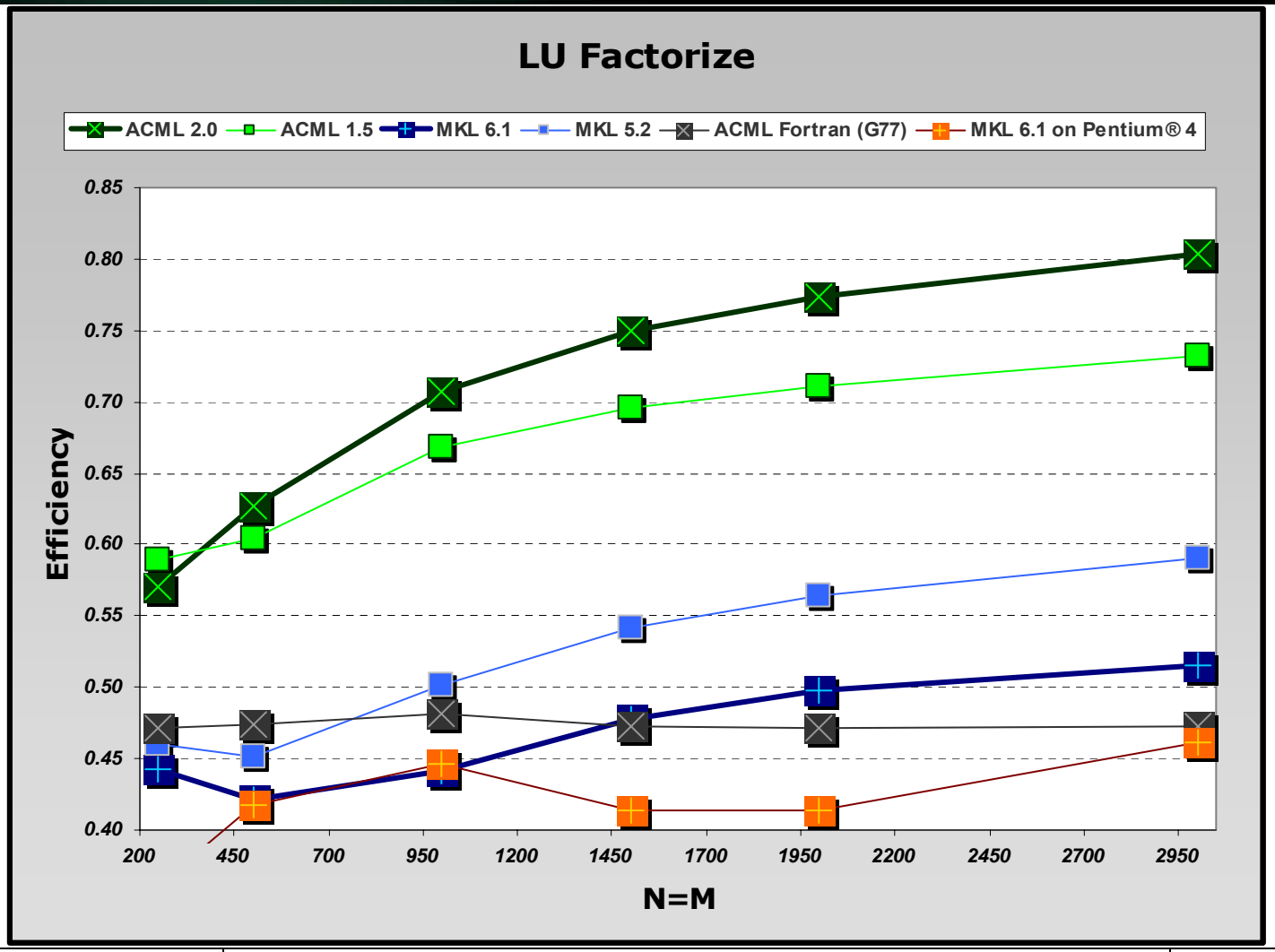
64-bit LAPACK Performance

SGETRF (Single Precision LU Factorization)



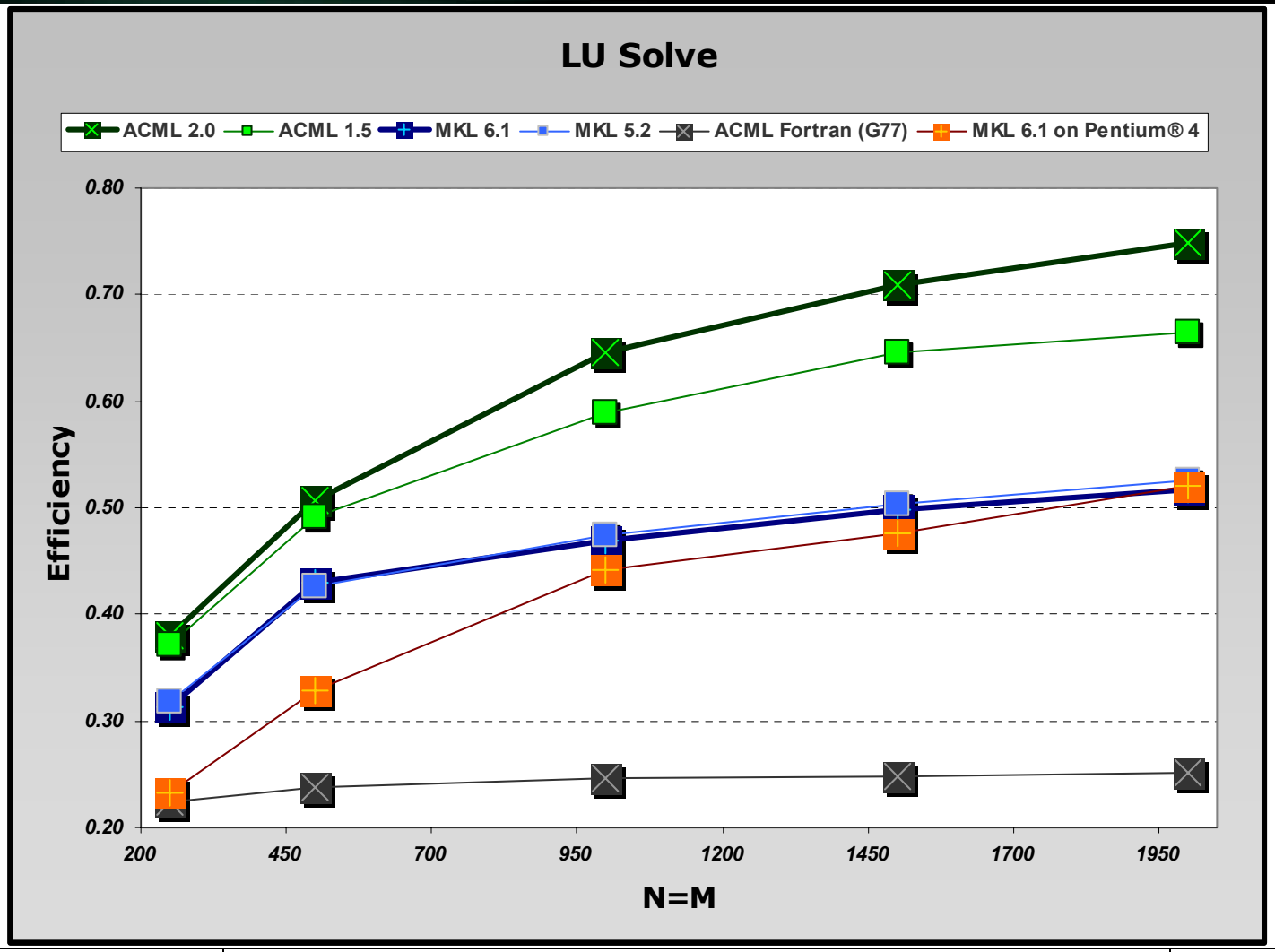
64-bit LAPACK Performance

DGETRF (Double Precision LU Factorization)



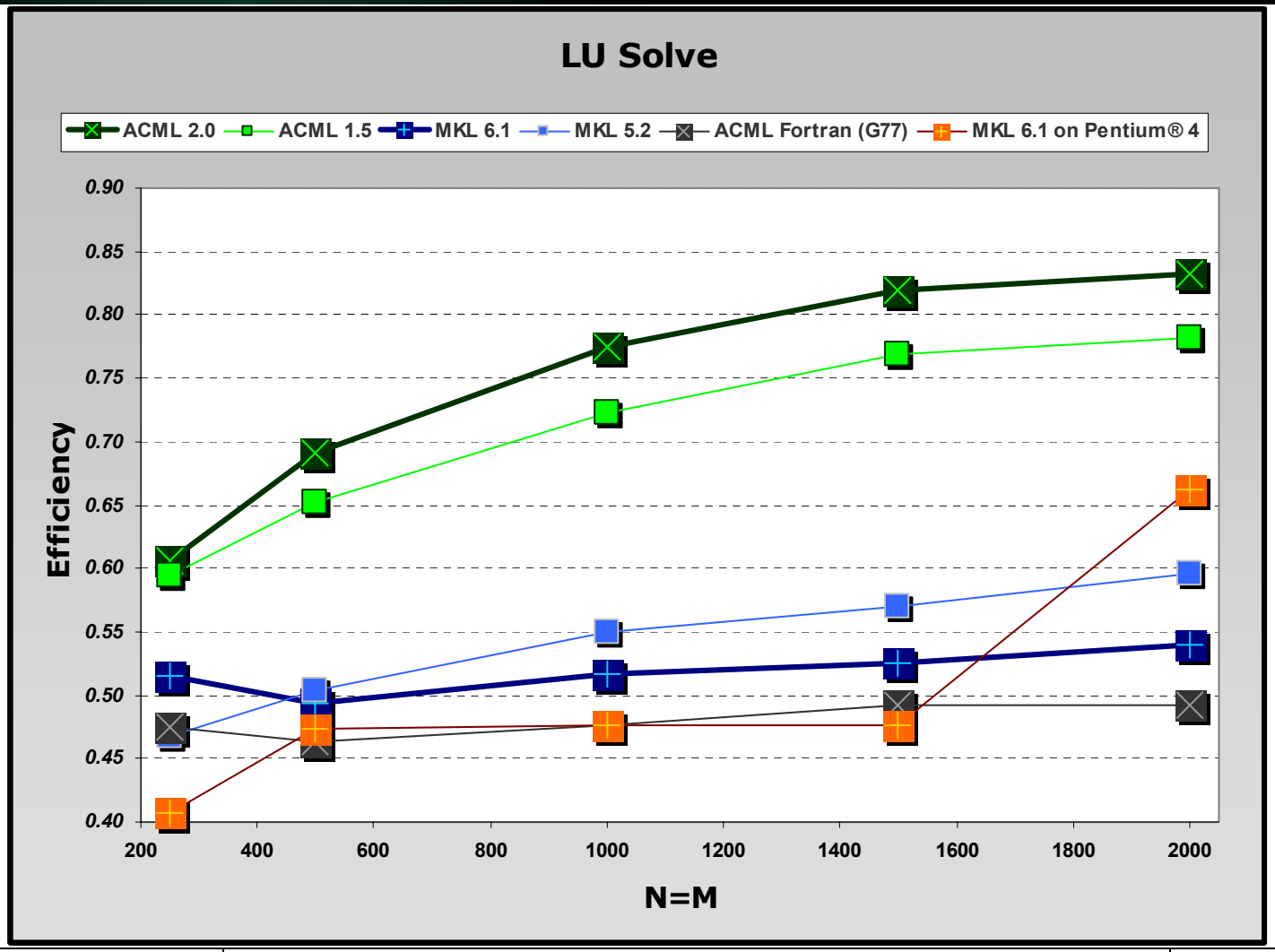
64-bit LAPACK Performance

SGETRS (Single Precision LU Solve)



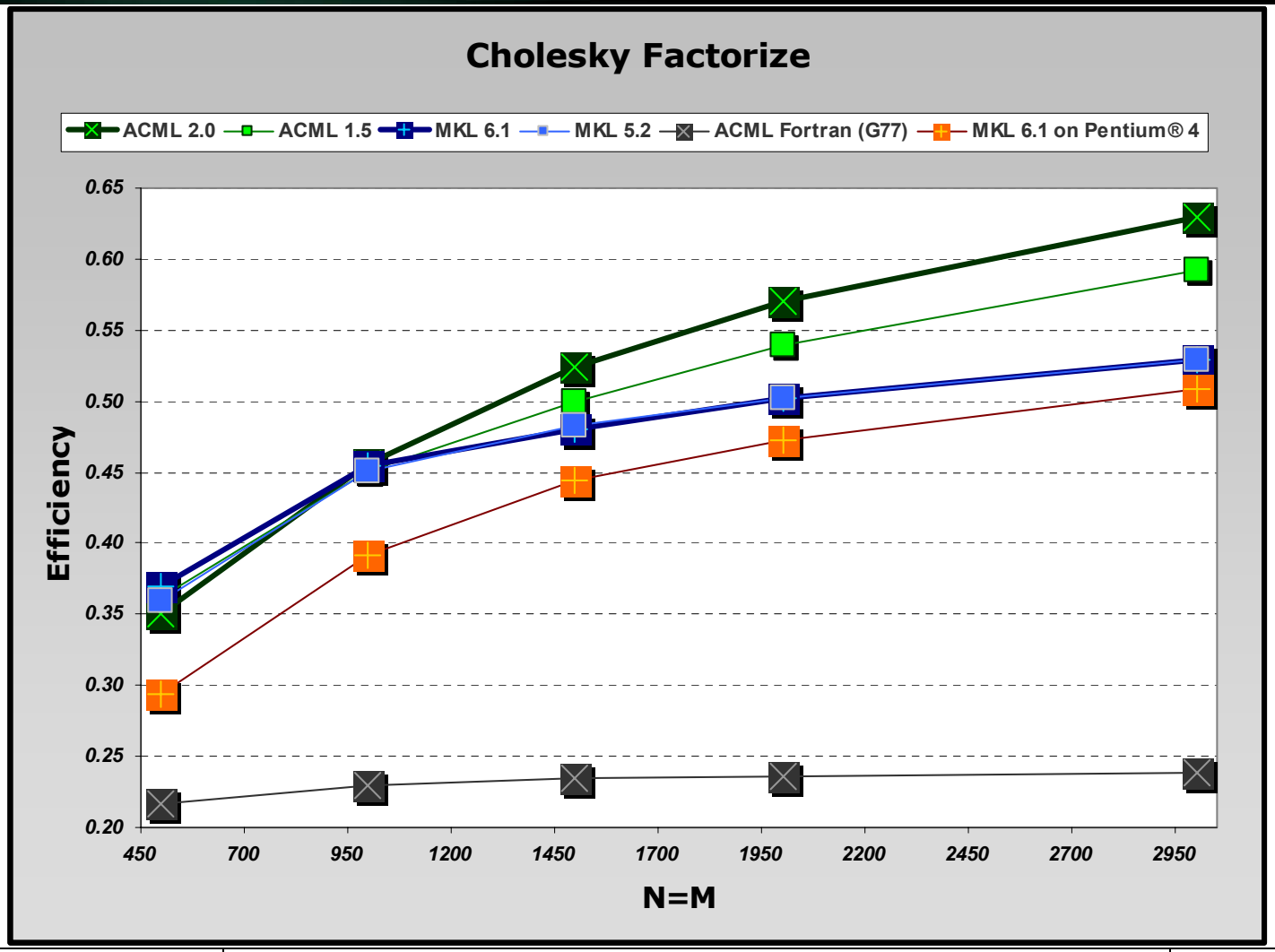
64-bit LAPACK Performance

DGETRS (Double Precision LU Solve)



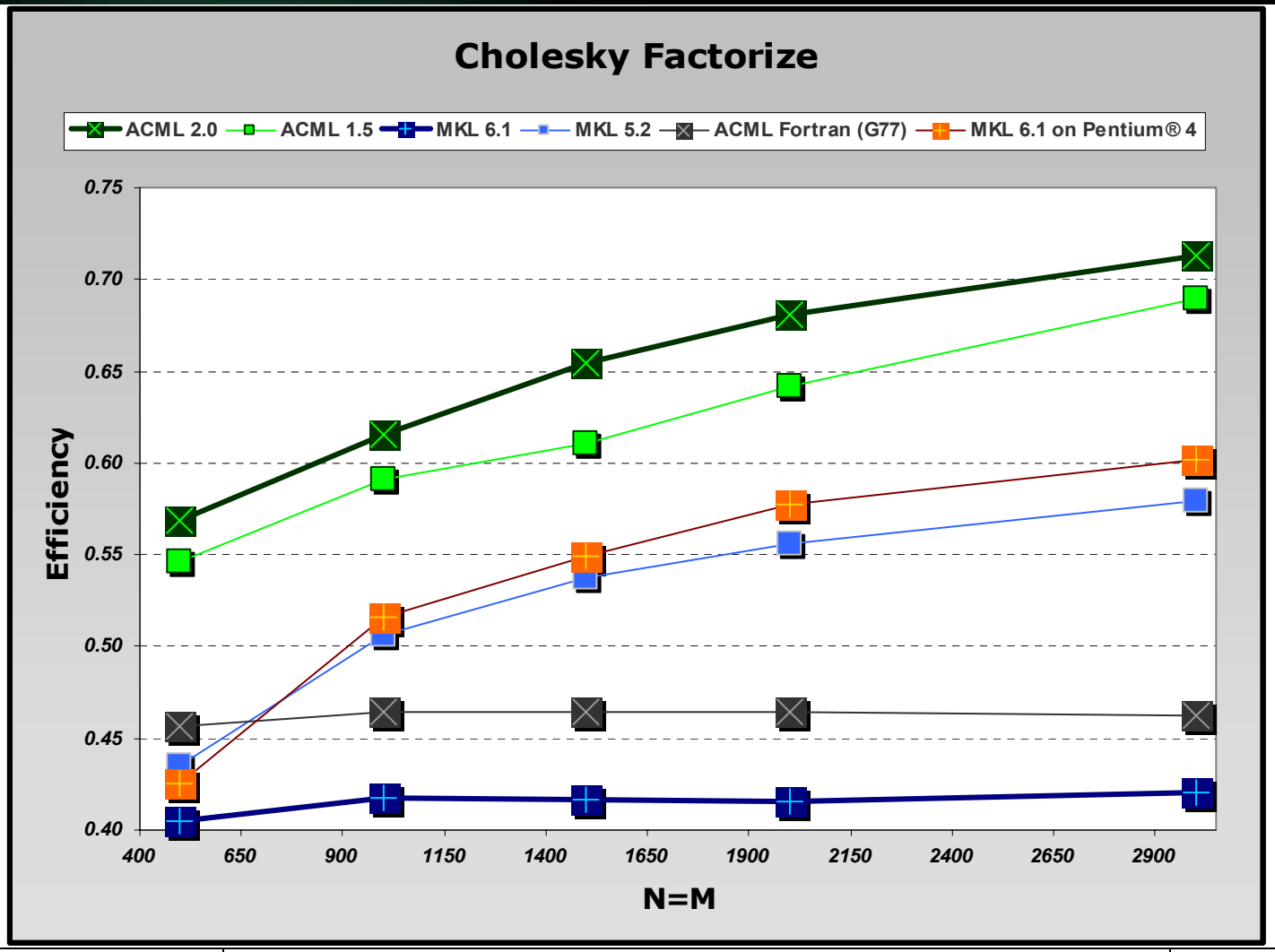
64-bit LAPACK Performance

SPOTRF (Single Precision Cholesky Factorization)



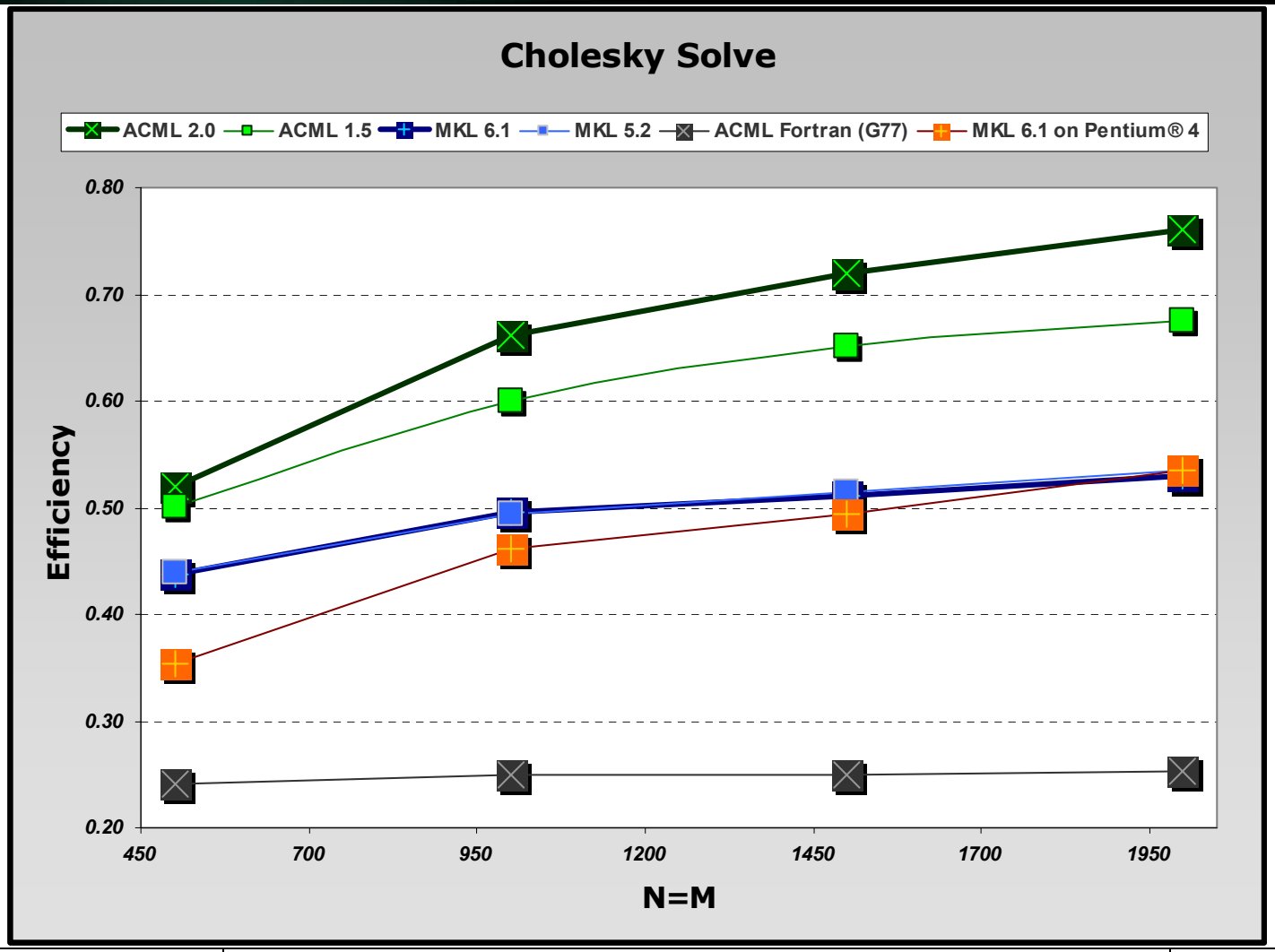
64-bit LAPACK Performance

DPOTRF (Double Precision Cholesky Factorization)



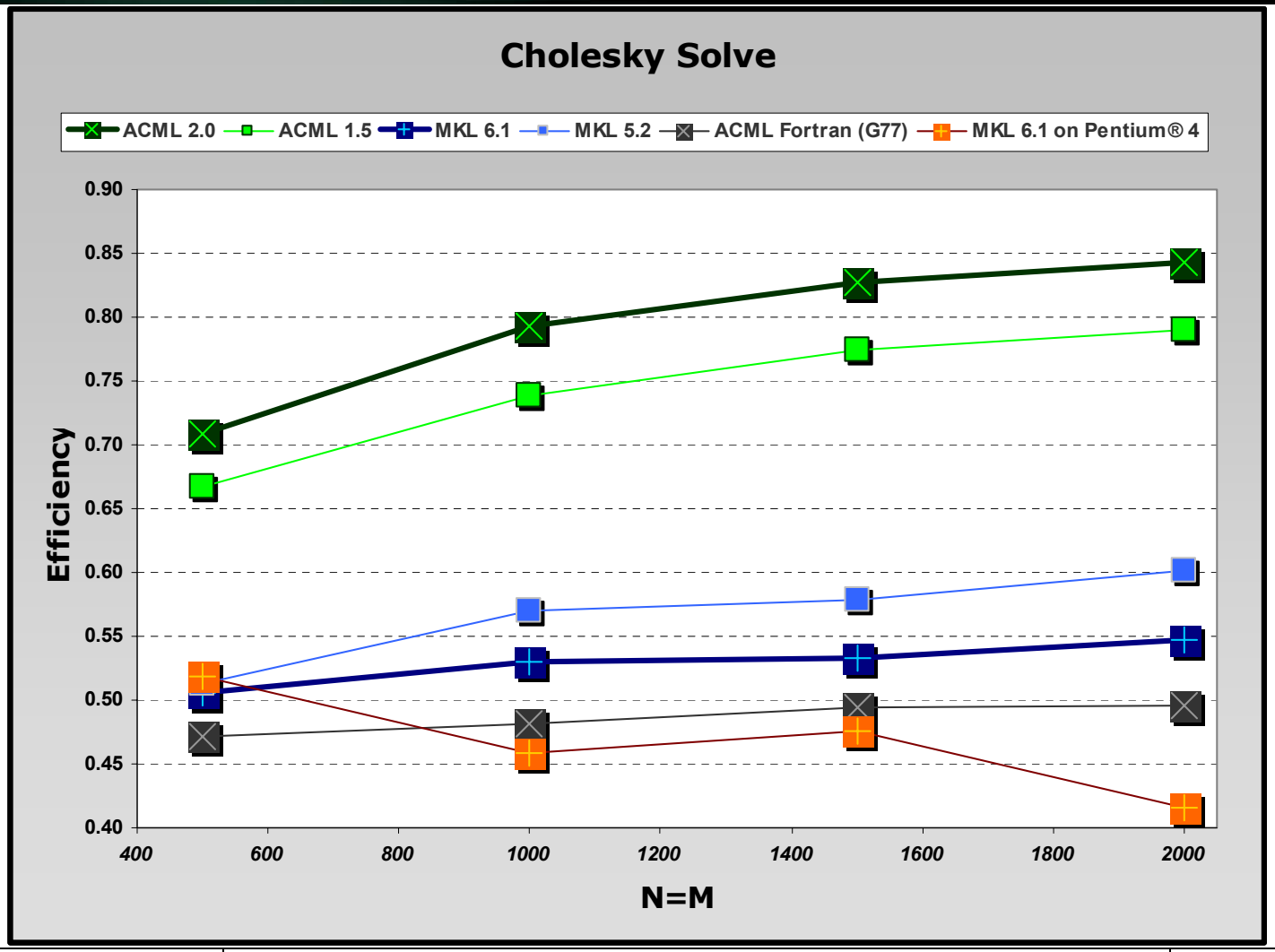
64-bit LAPACK Performance

SPOTRS (Single Precision Cholesky Solve)



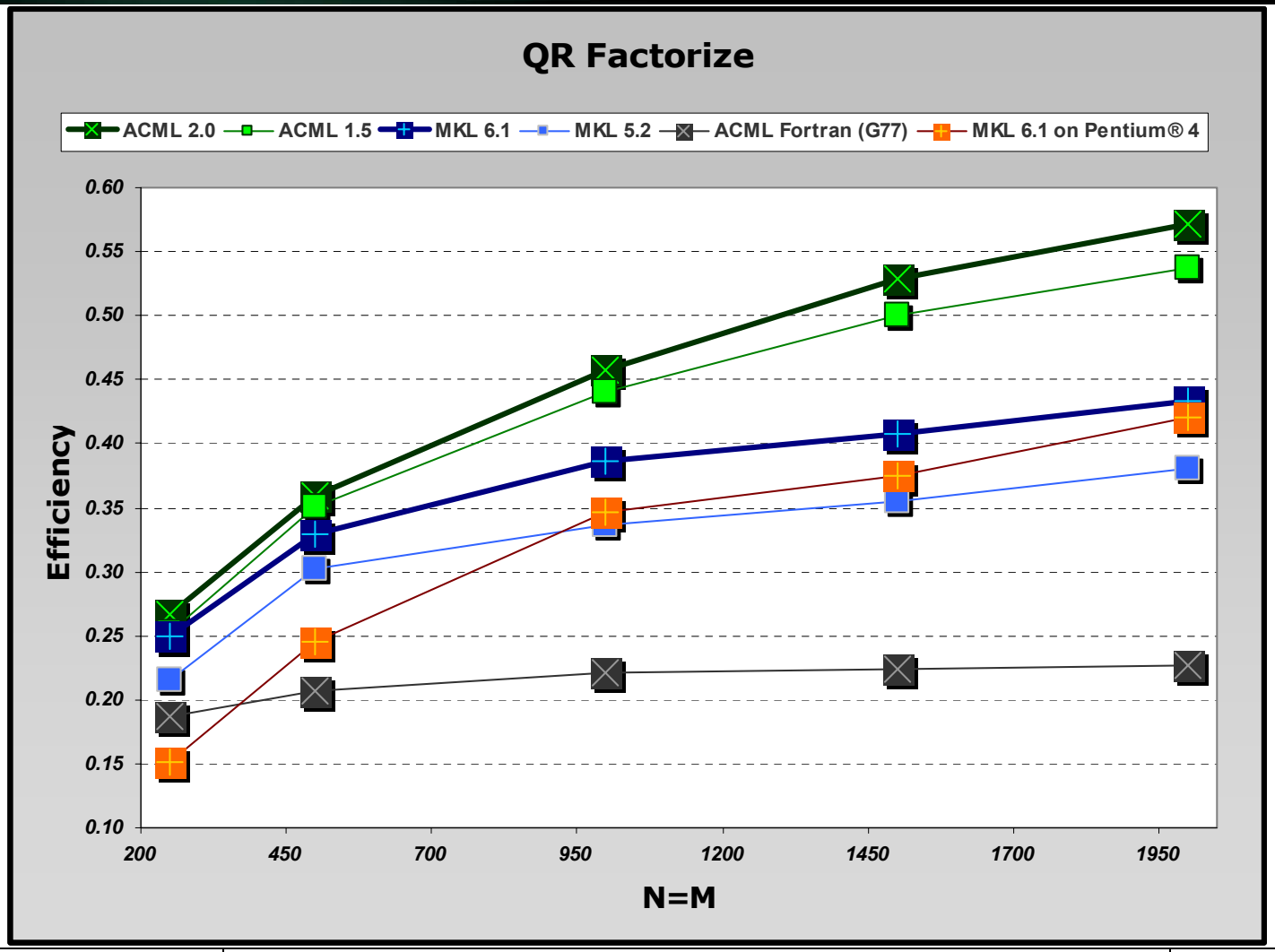
64-bit LAPACK Performance

DPOTRS (Double Precision Cholesky Solve)



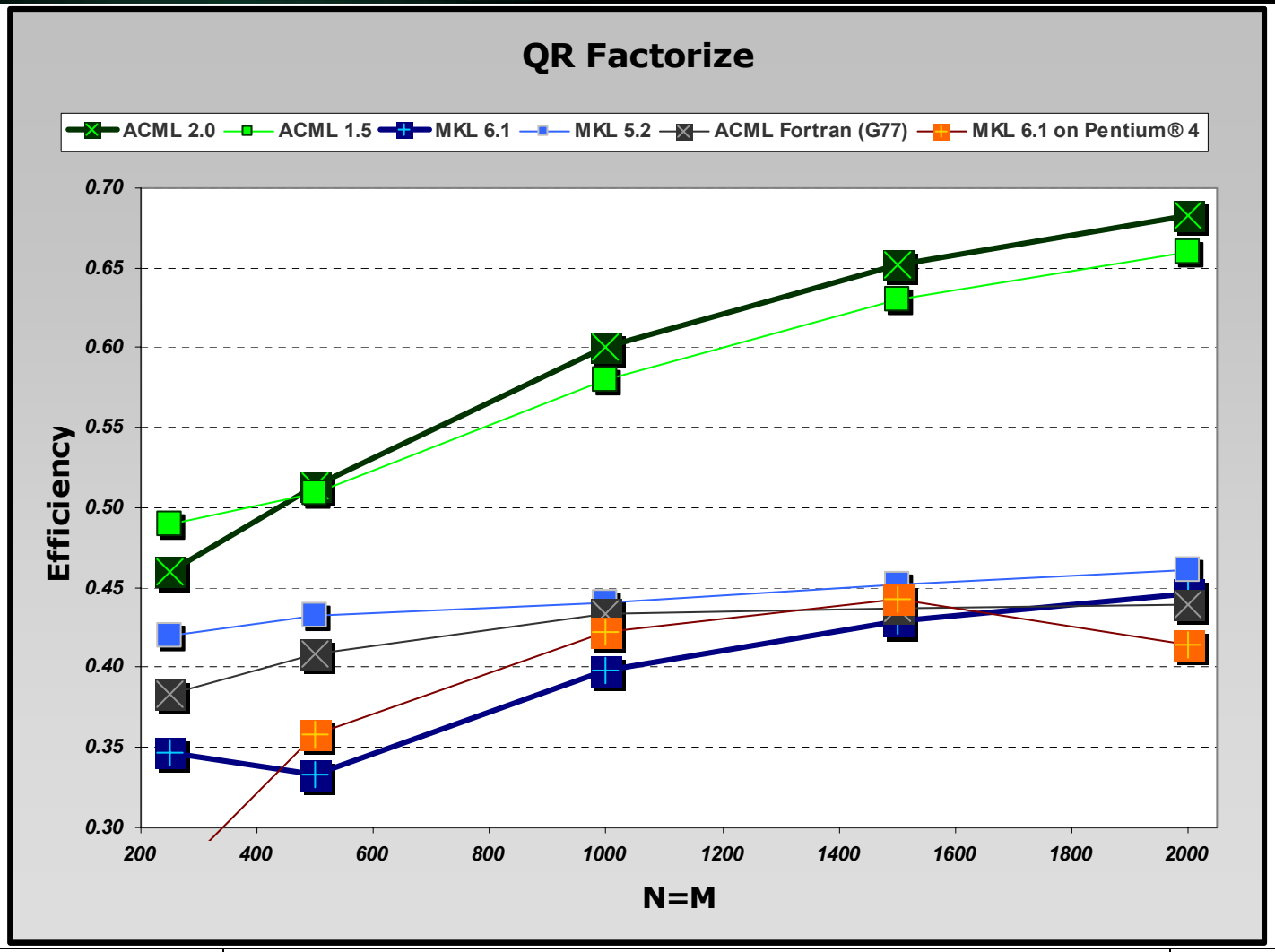
64-bit LAPACK Performance

SGEQRF (Single Precision QR Factorization)



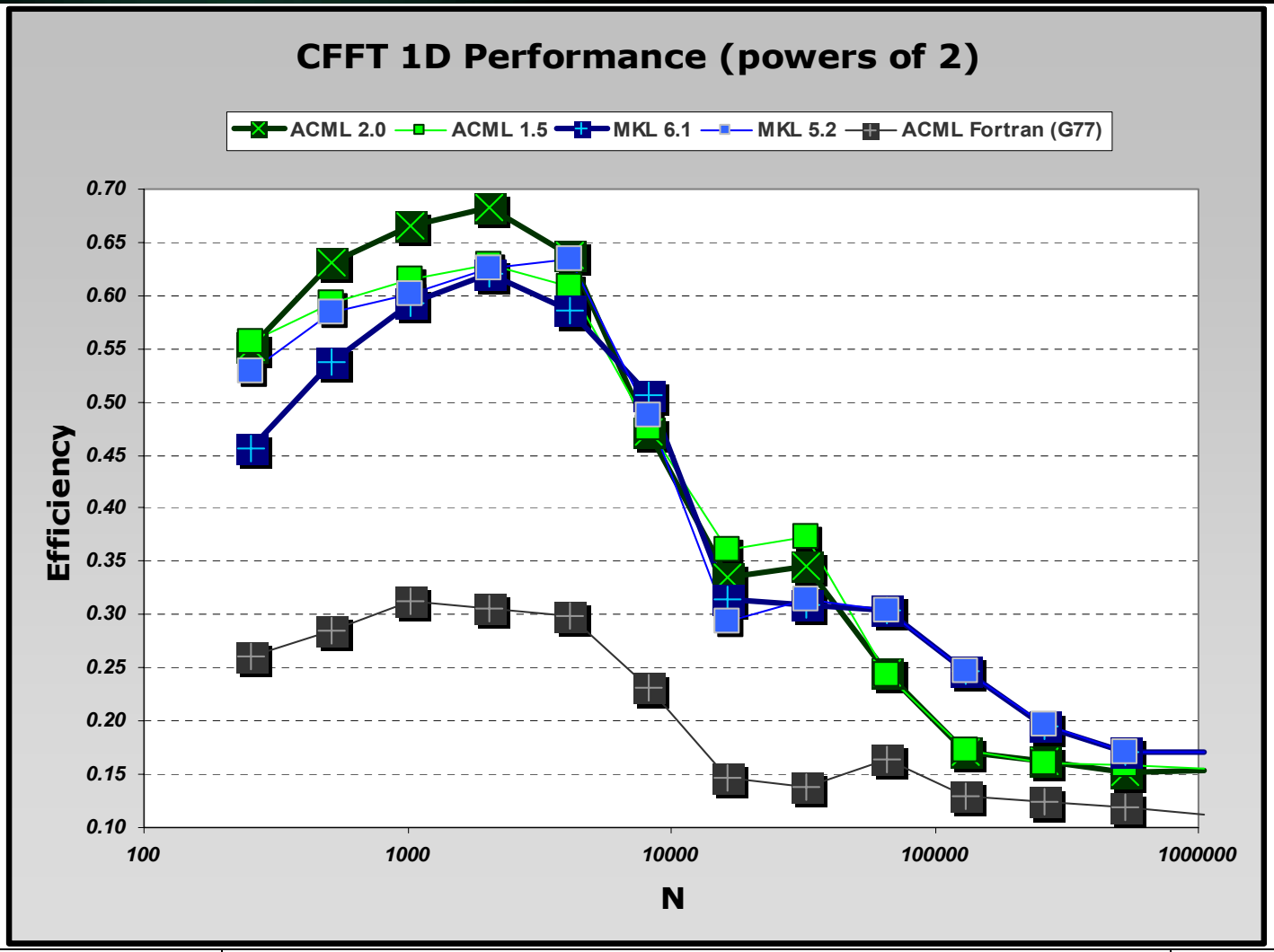
64-bit LAPACK Performance

DGEQRF (Double Precision QR Factorization)



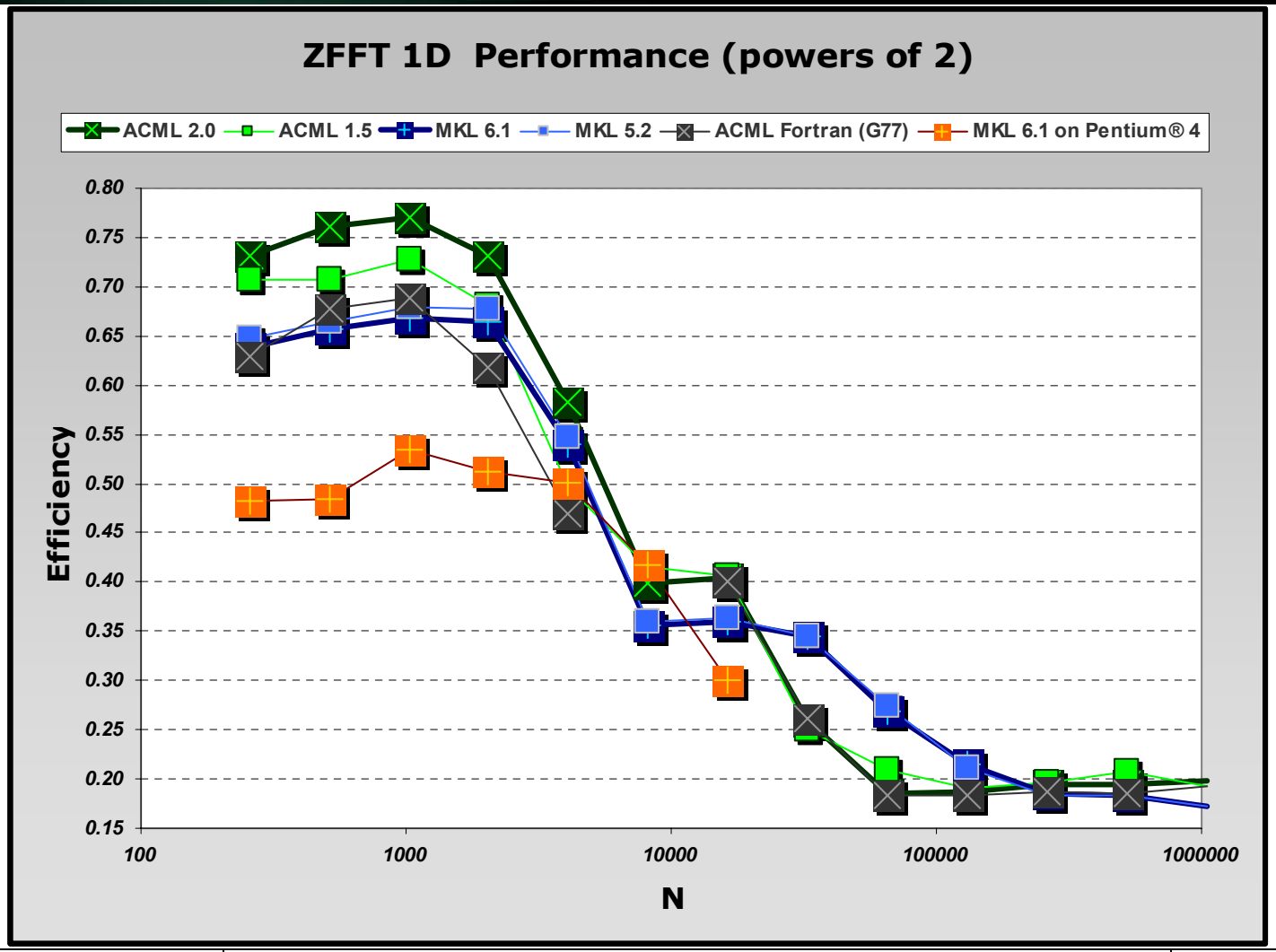
64-bit FFT Performance (power of 2)

CFFT1D (Single Precision Complex-Complex 1D FFT)



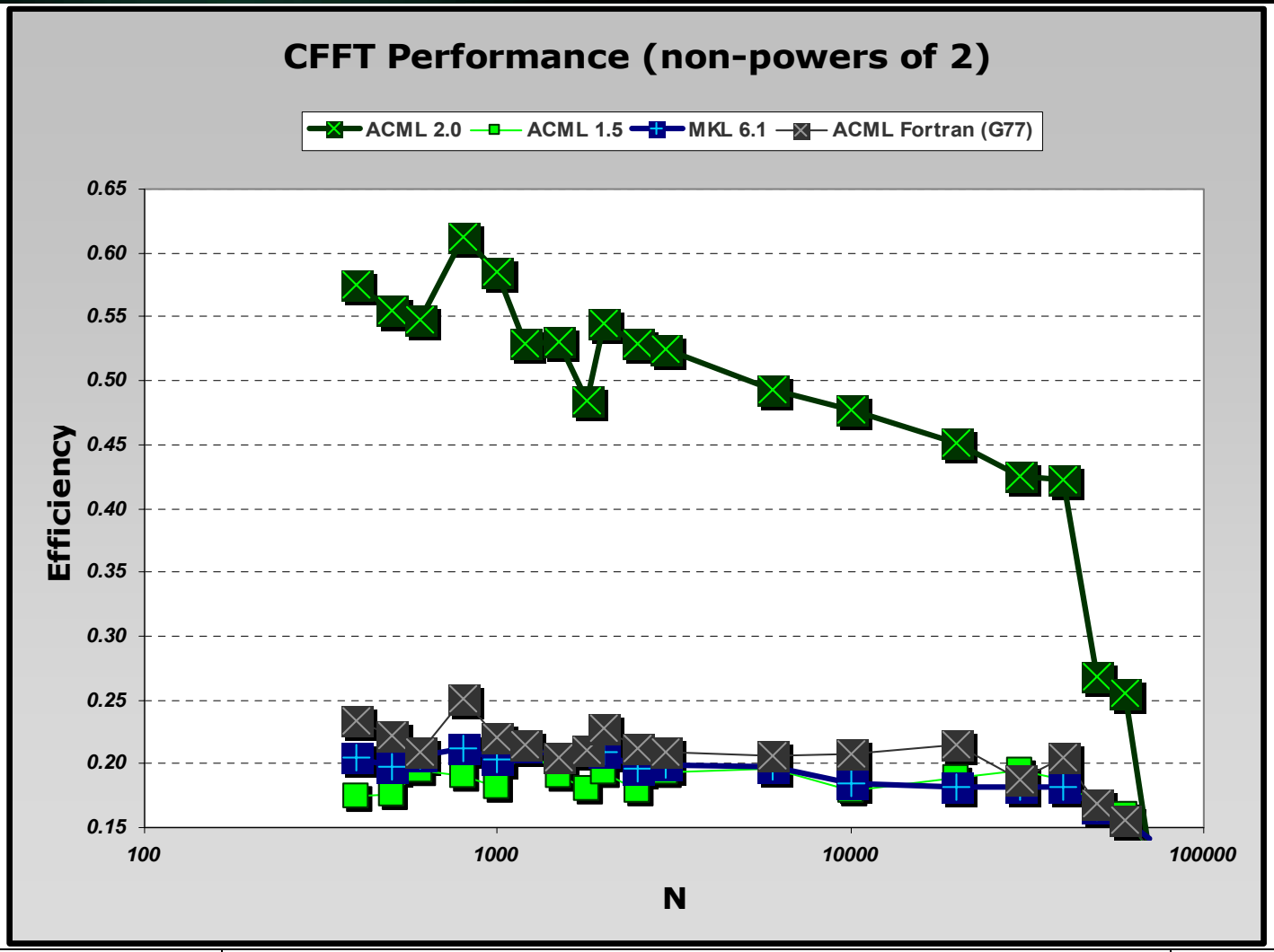
64-bit FFT Performance (power of 2)

ZFFT1D (Double Precision Complex-Complex 1D FFT)



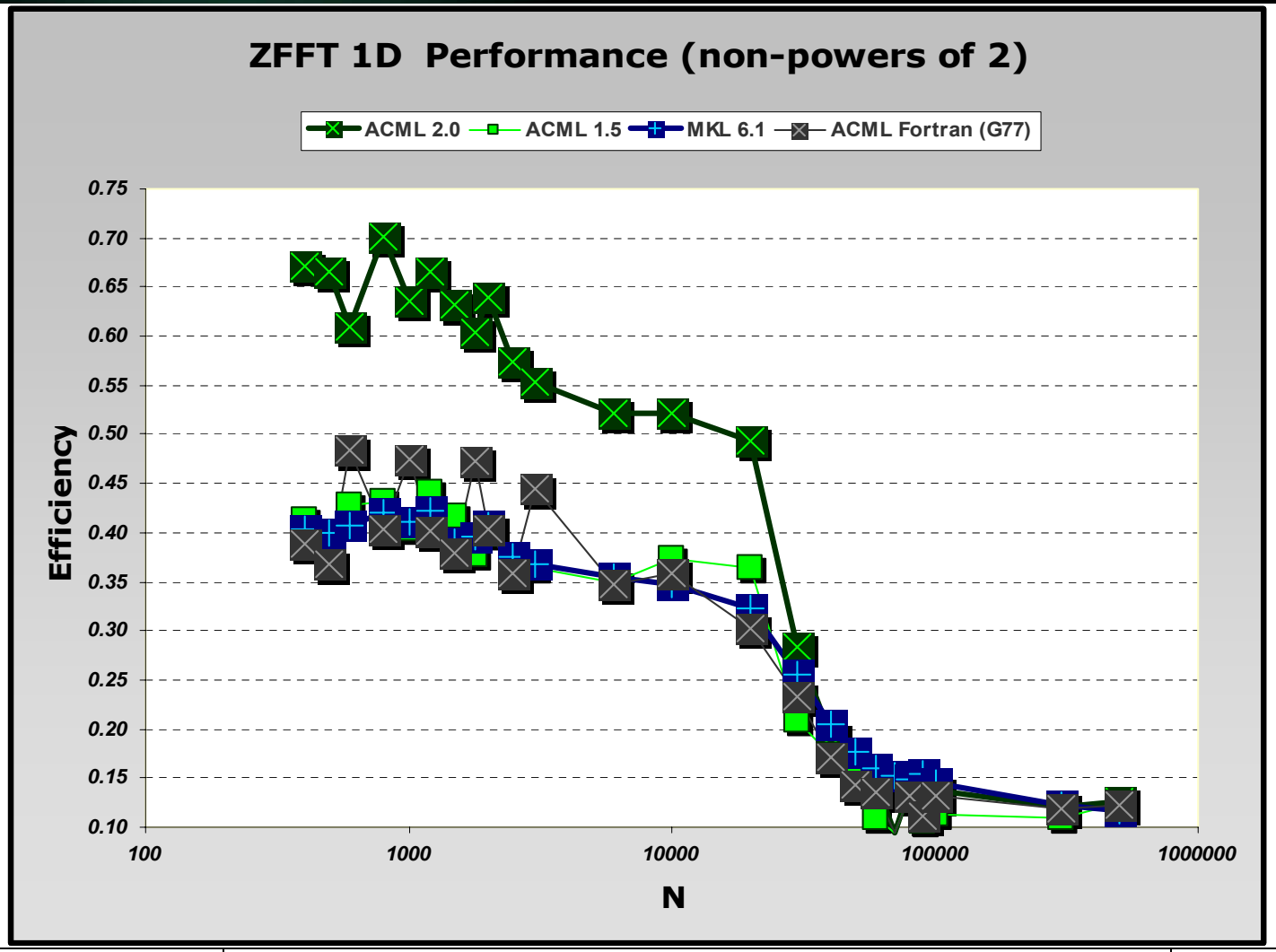
64-bit FFT Performance (non-power of 2)

CFFT1D (Single Precision Complex-Complex 1D FFT)



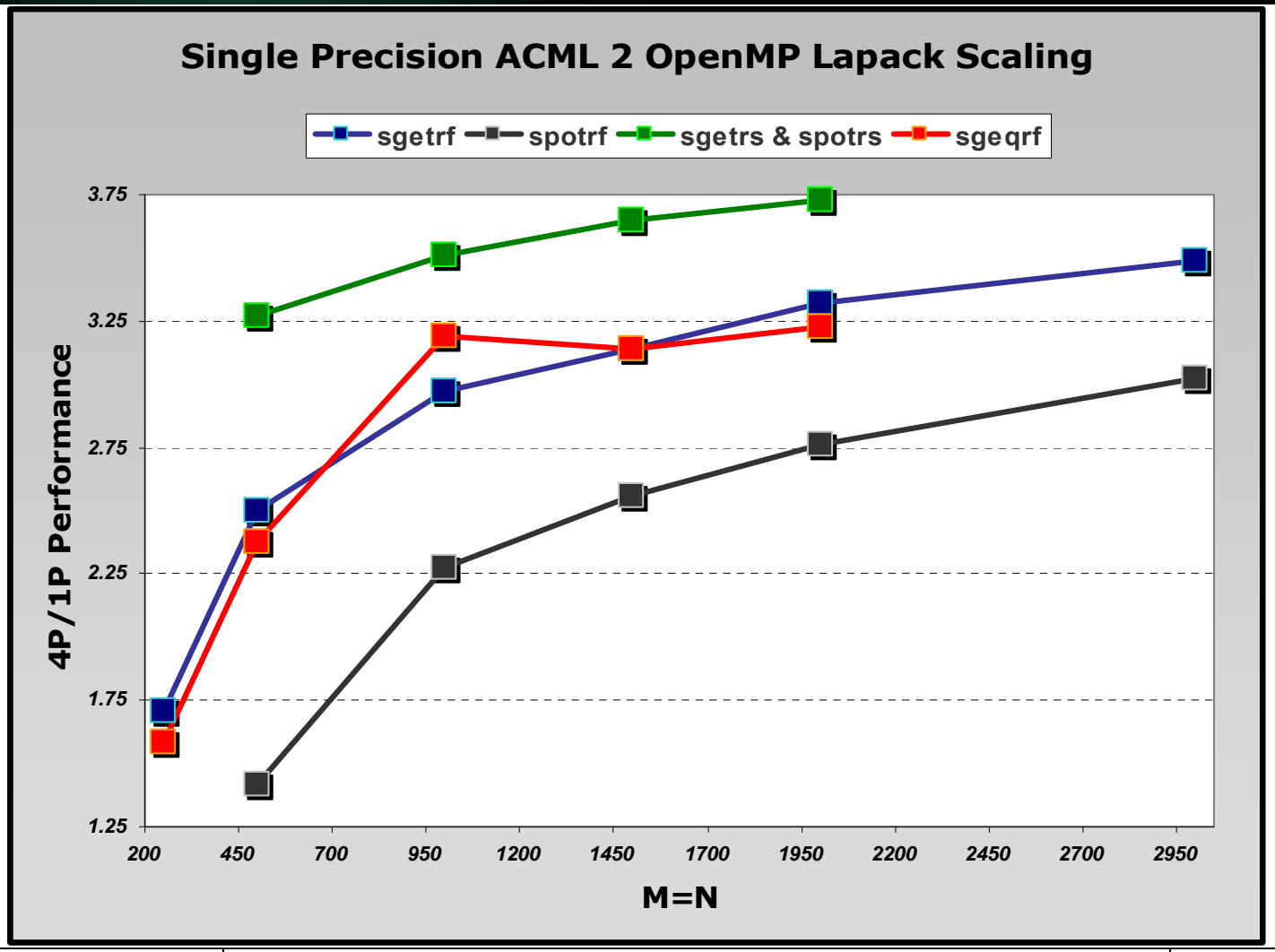
64-bit FFT Performance (non-power of 2)

ZFFT1D (Double Precision Complex-Complex 1D FFT)



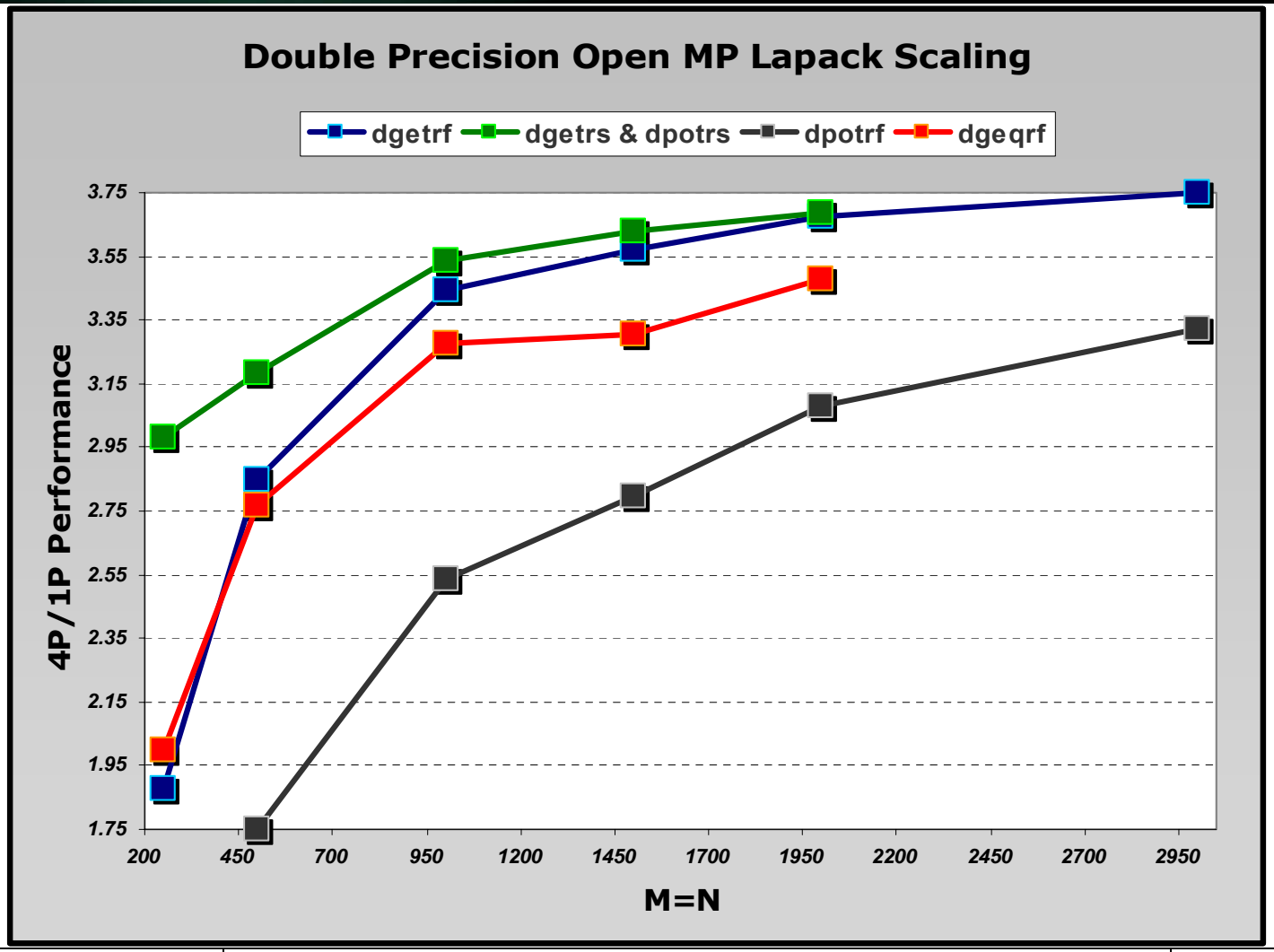
Multithreaded LAPACK Performance

Single Precision (LU, Cholesky, QR Factorize/Solve)



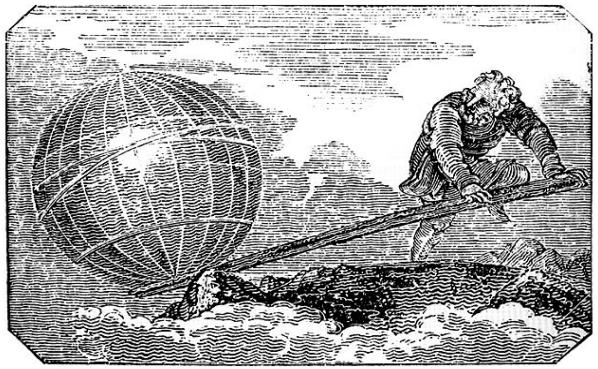
Multithreaded LAPACK Performance

Double Precsion (LU, Cholesky, QR Factorize/Solve)



Conclusion and Closing Points

- **How good is our performance?**
  - *Averaging over 70 BLAS/LAPACK/FFT routines*
  - *Computation weighted average*
  - *All measurements performed on an* **4P AMD Opteron™ 844 Quartet Server**
  - **ACML 32-bit** is **55% faster** than **MKL 6.1**
  - **ACML 64-bit** is **80% faster** than **MKL 6.1**



"Give me a long lever and a place upon which to stand and I will move the world " Archimedes circa 250 B.C

Test System Configuration
AMD

- **1P AMD Athlon™ 64 processor-based system**
  - **Processor**: *AMD Athlon™ 64 3200+ Processor*
  - **Motherboard**: *1P Solo AMD Development Platform*
  - **Hard Drive**: *40 GB IDE* 7200 RPM
  - **Memory**: *2 x DDR 2100 – 512 MB*
  - **OS**: *SUSE SLES-8 SP 3 64-bit Enterprise Linux*

- **4P AMD Opteron™ 844 processor-based system**
  - **Processor**: *AMD Opteron™ 844 Processor*
  - **Motherboard**: *4P Quartet Development Platform*
  - **Hard Drive**: *40 GB IDE* 7200 RPM
  - **Memory**: 4 x *2 x DDR 2700 – 4096 MB*
  - **OS**: *SUSE SLES-8 SP 3 64-bit Enterprise Linux*

Test System Configuration
Intel

- **Intel Platform**
  - **Processor**: *Intel Pentium® 4 Processor 1.7 Ghz*
  - **Motherboard**: *Intel D850GB*
  - **Hard Drive**: *40 GB IDE* 7200 RPM
  - **Memory**: *2 x 400 Mhz RDRAM – 512 MB*
  - **OS**: *Red Hat Linux 9 (Shrike) / Kernel Revision 2.4.20-8*
  - ***MKL** libraries Linked to:*
    libmkl_lapack32.so
    libmkl_lapack64.so
    libmkl_p4.so
    libvml.so
    libmkl_vml_p4.so
    libguide.so
    libpthread.a

Trademark Attribution